ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/221584033

A Game Prototype for Basic Process Model Elicitation

Conference Paper · November 2009

DOI: 10.1007/978-3-642-05352-8_17 · Source: DBLP

CITATIONS
7

READS
249

2 authors, including:

Stijn Hoppenbrouwers
Hogeschool Arnhem and Nijmegen
157 PUBLICATIONS **2,680** CITATIONS

SEE PROFILE

All content following this page was uploaded by Stijn Hoppenbrouwers on 18 October 2016.

The user has requested enhancement of the downloaded file.

A Game Prototype for Basic Process Model Elicitation

Stijn Hoppenbrouwers and Bart Schotten

Radboud University Nijmegen
Institute for Computing and Information Sciences
Heijendaalseweg 135
6525 AJ Nijmegen, the Netherlands
stijnh@cs.ru.nl, b.schotten@student.ru.nl

Abstract. We present a first prototype of a simple "modelling wizard". We also explain the ideas and rationales behind it: a first exploration of a new type of modelling tool which uses game-like interaction to guide and support the modeller in the process of modelling. After being played, the prototype game renders the basic information for a formal process representation (for example in BPMN), based on structured input given by a domain expert as she plays the game. Rather than offering substantial support for real modellers at this point, the game merely aims to demonstrate what we believe to be a new direction in thinking about methods and support for enterprise modelling. We also report on our experiences and evaluation of the prototype.

Keywords: Method engineering, process modelling, serious games, interactive modelling.

1 Introduction

This paper aims to contribute to the field of methods for business/enterprise modelling. Our work is related to methodological work in information systems analysis and design, and to the sub-field of method engineering [1]. Recently, we have proposed a somewhat alternative approach to the study and development of methods for *operational* modelling [2], meaning that we take the *practice* of modelling as our object of study, emphasizing the detailed actions and interactions that constitute the process of modelling. This contrasts mainstream method engineering, which mostly focuses on the definition of meta-models and high-level phasing of the modelling process, typically in terms of the required creation of various interrelated deliverables, without considering how every single element in such deliverables gets into place.

Our interaction-oriented approach to modelling research aims to be complementary to the mainstream approach in that it addresses issues hardly addressed in the mainstream, like aspects of modelling concerning human-human interaction, human-machine interaction, motivation, strategy and tactics, collaboration, decision making, negotiation, problem solving, and so on. We believe that understanding such aspects is crucial in better understanding requirements and enabling support for operational modelling processes, mainly with respect to the following points:

A. Persson and J. Stirna (Eds.): PoEM 2009, LNBIP 39, pp. 222–236, 2009.
© IFIP International Federation for Information Processing 2009

1. Improve its quality [3,4]
2. Improve its focus in view of its utility [5,6]
3. Improve its efficiency
4. Make "lightweight formal modelling" more accessible to non-expert modellers (i.e. enable modelling without the necessary presence of facilitators or expert analysts; this is sometimes called "disintermediation"[7])
5. Improve the possibility to perform (formal) modelling in a truly collaborative setting

The initial results presented here emphasize the fourth point, but also involve the first, second, and third point. We do not address the fifth point. Also, we do not (yet) attempt to provide any solid proof that our approach increases quality, focus, and efficiency of (some aspect of) enterprise modelling. However, we do claim to provide at least a reasonable proof of concept and a demonstration of the possibility and potential of creating a game-like, modelling process-oriented type of modelling tool that goes beyond mere editors.

The sort of model that results from applying the approach presented is a basic process model of the kind usually represented in schema languages like UML Activity Diagrams [8], BPML [9], or YAWL [10], featuring basic concepts like activities, flows, and AND/OR-splits. However, we deliberately refrain from tying ourselves down too much to some specific language for process modelling, because:

- We want to avoid discussion here about detailed differences between languages and their merits, for reasons of focus;
- We observe that in practice, initial phases of process modelling do not normally call for detailed decisions concerning language/representation (whereas later stages often do);
- We aim to primarily elicit *information on the basis of which a schematic model can be generated* rather than needlessly confronting the participant with an actual schema, in particular in the earlier stages of the modelling process (which are our main focus). This does not exclude the use of schematic representations, but the utility thereof is different: they serve to extract information; they do not necessarily constitute the actual final model.

We have chosen to focus on process modelling here because we believe it is the most central type of modelling in most enterprise modelling efforts. Even so, the approach presented already includes some aspects typically related to basic data modelling or ontological modelling. In fact, we work towards the future operational integration of elicitation of (at least) (Business) Process Modelling, Ontological or Data Modelling, and (Business) Rule Modelling, and believe such models can indeed be fruitfully created in parallel, i.e. one model aspect can provide helpful (if not vital) information for the creation and validation of another. This principle will be pivotal in our wider (longer term) approach.

As mentioned, we propose to embody modelling methods in games. As discussed in [2,11], this minimally requires the following elements to be present:

- A clear description of the task to be completed by the player(s), including a victory or end condition for the game;
- A clear description of the game components to be manipulated by the player(s)
- A clear description of the rules to be followed in carrying out the task (i.e. playing the game);
- A clear description of the allowed types of action and interaction within the game;
- A clear description of procedures to be followed by the game system (for example, keeping or calculating a score)

Goals set within the formal context of a game correspond to task descriptions or assignments (what we will call "game-internal goals"). Such goals should not be confused with any goals the playing of the game is supposed to fulfil: its utility goals ("game-external goals"). For example, though our external goal for the game presented is to create a basic process model, the player is not explicitly assigned the task to do this (the terms used instead are "task description", by means of "describing task steps and their ingredients and products").

Game-external goals are explicitly linked (in design) to any desired properties of the resulting model, or even mind-states of modellers and –in collaborative games–social effects (shared understanding, agreement, commitment). Relevant goal categories have been discussed at some length in [12,13], and include utility goals, deliverable goals, validation and agreement goals, syntax goals, interpretation goals, argumentation goals, and efficiency goals.

Designing the goals, rules, score system etc. in line with game-external goals and standing conventions (explicit or implicit) is by no means trivial. In fact it is the main long-term goal of our line of research to discover and develop apt sets of interaction rules for achieving specific external goals in view of different capabilities and expertise of players involved, and of different demands posed by the modelling domain and context. The current game reflects only a very first (yet concrete) exploration of the basic principles.

2 Utilitarian Idea behind the Game

It may be helpful for the reader to view the game as a preliminary design for a "process modelling wizard". We are fully aware that some experienced modellers, or even "not-so-experienced modellers", find the current game restrictive and somewhat tedious. However, this would not necessarily render the game presented useless: its purpose is merely to make a start in creating games embodying playful modelling procedures, opening up process modelling for layman modellers, and to illustrate the more general point that shaping modelling methods as game designs is possible, interesting, and *potentially* useful.

The game in an operational sense works towards a particular type of *result*. We will elaborate on this now. As discussed, creation of an actual process schema is neither the game-internal nor the game-external goal of the game as such: rather, that goal is to *deliver information* that can be directly used to *derive* a process model. In order to obtain some particular sort of structured information (a "pre-model"), in

some cases additional information is needed which is not necessarily to be reflected in the final model. We emphasize that this does not mean such information is irrelevant to the modelling *process* (in particular, the thinking process). In fact, we believe that quality-driven, stepwise elicitation of the sort we try to realize in the game *requires* elicitation and conceptualization of knowledge that supports *thinking about aspects underlying the process* in the domain rather than merely the abstract representation of that process in some focused but therefore also restricted modelling language.

For example, in our game we enforce the definition of objects and attributes, making the player provide what can be taken as a structured argumentation for the use of standard AND-joins (also see [12]). For example, as illustrated in the middle column of Fig. 1, a business process model in the standard language BPMN [9] typically shows an ordering of activities, e.g. activities D and E must be completed before activity F can be started. However, the reason why this is the case is that D and E respectively produce entities n and o that are needed in F (resulting in what is technically called an "AND-join"). This is illustrated by the text in the leftmost and rightmost columns of Fig. 1, in which these entities and dependencies are made explicit. However, such dependencies and entities are not made explicit in a regular BPMN diagram, even if they are crucial for creating a useful, "good" one. As a consequence, the entities and dependencies involved are usually left implicit and exist only as concepts in the head of the modeller –in fact, they are probably more concrete to the modeller than the abstract process flow derived from them. Even if the objects in the process are made explicit, perhaps in another model, they are not explicitly used as a basis for deriving AND-joins.

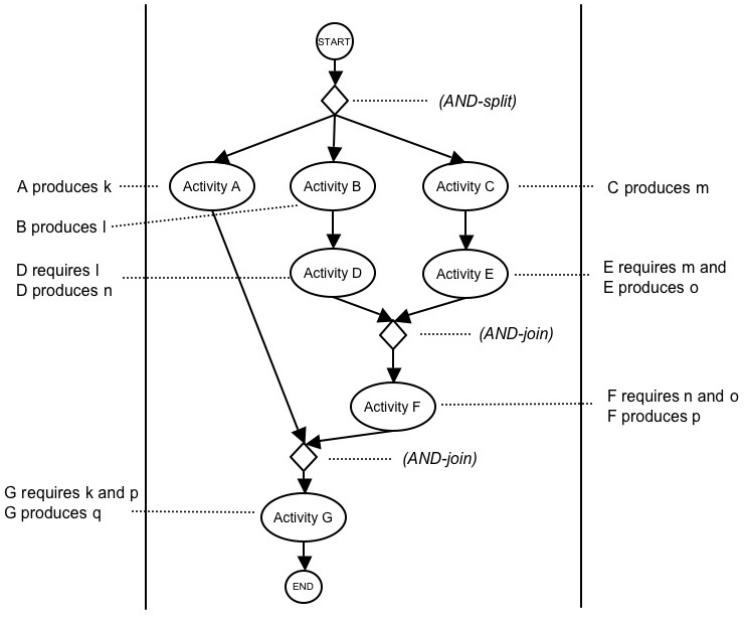

Fig. 1. Reasoning about Basic AND-joins

Put succinctly, the immediate utility goal (i.e. game-external modelling goal) in our game is to put the above argumentation central and strive to *indirectly elicit basic BPMN-like structures with AND-joins*. Note that OR/XOR-joins are excluded for now.

3 The Game

We will first provide an illustrated overview of a simple but typical course of gameplay. Following [2] and [11], we will then give a brief decomposed overview of the game design along the lines of Game Design Theory. Please note that we feel the gameplay as described below does insufficient justice to the actual game experience. The static pictures look very much like those one might expect from a regular graphical editor. We emphasize the difference lies in the act of modelling as such; for this, a real demo or, even better, actual experience in playing the game would be required. Also note that the example presented is meant to explain the main game mechanics reflecting the rationale pattern presented in the previous section. It is a toy example that was also used in the tests for the initial, exploratory game round, but is simpler than most assignments used for testing the game.

The external (utilitarian) purpose of the game is to get the player to describe a basic task accurately, in terms of its *steps* (which is the term the game uses for 'activities'). The game looks somewhat like a normal modelling tool, but provides more guidance for stepwise thinking and (most importantly) does not require abstract thinking about AND splits and joins: it merely inquires about what is needed for a step (i.e. *ingredients*), what items come out of it (i.e. *products*), and (optionally) what change is inflicted on some item in the course of the step (a *link*). The game-like properties of the procedure (please be aware of the fact that it is, after all, a methodical procedure dressed up as a game) are the following:

1) The game can only be finished if the player fulfils a minimal set of demands, because only then the required information can be derived;
2) The player gets immediate feedback on what she is doing, using graphics and sound; also, a score is calculated and made visible;
3) How long the player plays is reflected in the score, while the time is visibly ticking away, thus introducing mild time pressure;
4) The player is (hopefully) motivated or entertained by the setup and gameplay, besides being guided.

The player has the option of being shown extra (rather minimal) guiding and explanatory remarks, meant to help novice players understand the game and not miss some finer points. In the example pictures, we have excluded these (i.e. switched them off), except in the first illustration (Fig. 2). In addition to the optional guidelines, hints are also shown on the bottom of the screen when the player moves his mouse over any object (standard). This is not visible in the figures.

At the beginning of the game the interface provides the player with only one button, allowing the player to create a new step. Other than that the player must give the task a name. When a new step is created it appears as a rectangle that the player can drag around; the next thing to do is to give the step a name (Fig. 2). A guideline for this is to describe the step in no more than four words.

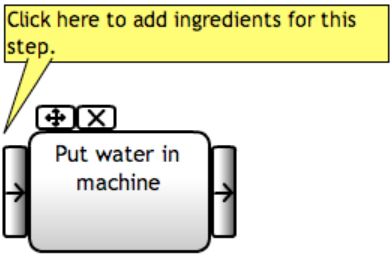

Fig. 2. Creating a step

Not visible in the illustration is the brief appearance of a green, animated number "10" drifting away from the activity symbol, indicating that 10 points have been scored by this action. There is also an accompanying sound.

Next, the player then has to add "items needed" (*ingredients*) and "items created" (*products*) to the step. Each ingredient that is added to a step also shows up in a list in the top left of the screen (Fig. 3). Items from this list can then be dragged to a new step, to be reused. This minimizes repetitive typing and provides a clear overview of items introduced so far. It also encourages re-use of exact terms.

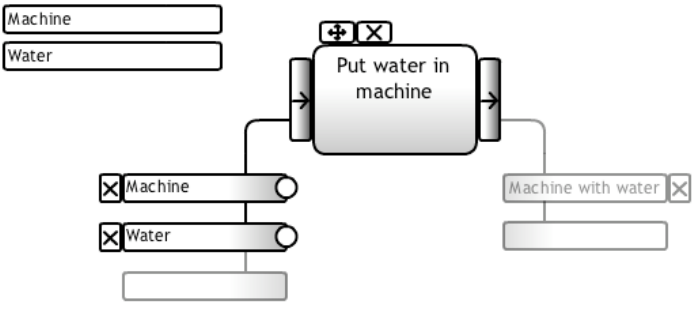

Fig. 3. Adding needed and created items

Next, clicking on a small bubble next to an ingredient and then clicking on a bubble next to a product of the same step can connect ingredients and products; again there is a (rather funny and appropriate) sound effect. This creates a link (Fig. 4). A link can be used to provide extra information about what happens to an ingredient during that step.

However, when an ingredient gets linked, the link should describe what happens to the ingredient. As a conceptual aid, the player may describe the change by filling in the pattern "this ingredient is being ...". The grammatical trick is that the player provides a verb that can also be used as an adjective, and therefore as an attribute of the item (describing a relevant state of it): a *precondition* (though this term is not used in communication with the player). This is useful because if the item is also used in another step, the game will recognize that the item has attributes, and provides the

player with the simple option to select one or more existing attributes as relevant to the step (illustrated as part of Fig. 5, "filter::put in machine").

Another option for describing a link (and its underlying precondition) is to combine two ingredients in a pattern known from data modelling: ingredient A [with ingredient B] & ingredient B [in ingredient A]. An extra option the player has here is to graphically connect the links, which automatically render this precondition description pattern (not illustrated).

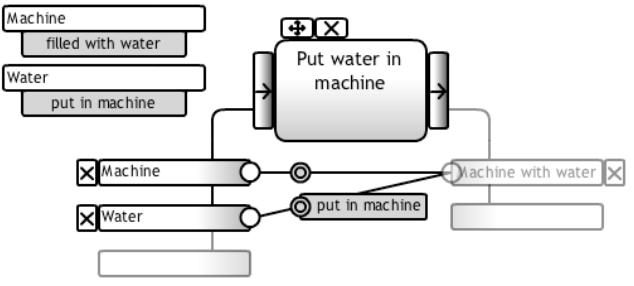

Fig. 4. Adding a state change attribute

When the player has described multiple steps this way, the game derives the connections between the steps. Each step has a set of ingredients with possible preconditions and a set of products as well as ingredients with added preconditions. When an ingredient of one step matches with a product of another step, the two are connected (Fig. 5), and a triumphant sound is played (taDAA!).

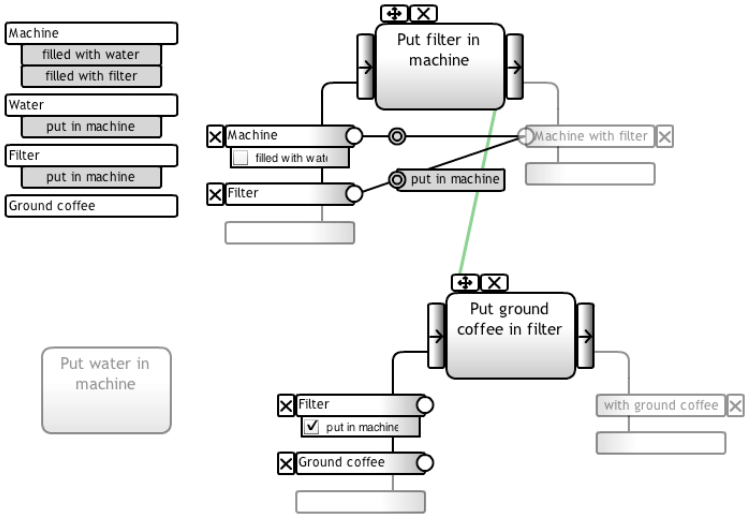

Fig. 5. A connection is found

Optionally, the game can automatically visualize the derived suggested order of connected steps by moving ('floating') them to a relevant position when they are not being used by the player. However, note that this does not amount to the visualization of flow as in an actual flow chart. A flow chart is derived from the above diagram later (outside the game as such).

The player can only finish the game when all the steps she created are connected to at least one other step, but can carry on until the model is complete; this is up to the player to decide (Fig. 6).

We will now briefly describe the main game components (objects to be manipulated in the game), game mechanics (actions allowed to take place on the components), and game rules (goal descriptions, constraints, score system). The rules are not described in great detail, as most of them have been demonstrated already.

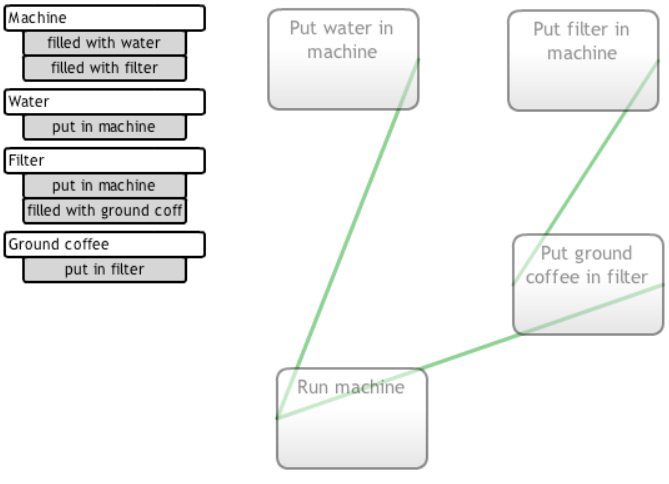

Fig. 6. Finished

3.1 Game Components

The game components are: Task Name field, Step Symbol with Step Description, Ingredient boxes and Precondition Ticks, Product Boxes, Link Circles and Boxes, Item/precondition List Boxes.

3.2 Game Mechanics

First, there are game mechanics for filling in various textual fields: Step Name, Step Description, Ingredients, Products, Link Descriptions. There are also non-textual mechanics: Creating Steps, adding and deleting Ingredients/Products, dragging Item List Boxes to Ingredient or Product Boxes, ticking Preconditions, linking Ingredients with Products.

Layout functions are left out here since they are auxiliary to actual gameplay. They belong to the game interface, which we further disregard here (but see the illustrations for a visual impression).

3.3 Game Rules: Goals, Assignments; End Condition

The (internal) goal of the game is to score points by creating a stepwise description of some (self-)assigned task, within possibilities and rules as embedded in and constrained by the interface. Obligatory: two interconnected steps (minimal end condition), implying at least one product matching one ingredient, so the game cannot be ended without some ingredients and products being entered. Importantly, the actual end condition is that the player herself "calls it quits" when the model is finished. The game as such does therefore not provide means to decide when the game is finished, only when it is not. Note that this is not unheard of in the Gaming world: many modern role playing video games can in principle be played ad infinitum.

3.4 Game Rules: Score System

The scores are calculated as follows:

- 100 points for each step.
- 100 points for each connection between two steps.
- 10 points for each ingredient, product or link.

Note that deleting a step leads to a reduction of the score with 100 points. The final score is the sum of the score so far minus half a point for every second played. However, the amount of points deducted based on the elapsed time can never be more than half of the amount of points scored.

3.5 After the Game: Deriving a BMPN Diagram

Besides the game as such, we also implemented a simple algorithm for deriving a BPMN-like structure from the information gathered. Instead of generating an actual diagram (Fig. 7), we decided that for this prototype an XML-based format, XPDL [14], would suffice.

```
#Start by finding the sets of input and output items for each step:

for each ingredient
    for each precondition
    if precondition is selected
      add "precondition_ingredient" to input set
    if no preconditions are selected
    add "ingredient" to input set

for each product
  if product is relevant
    add "product" to output set
for each link
```

Fig. 7. Algorithm to find dependencies between steps

```
  if link is not empty
    add "link_ingredient" to output set

# Whether a product is "relevant" (not overwritten by
# a link and not present as an ingredient) is known
# beforehand. A link always has a reference to its
# corresponding ingredient and product.

# Next, find the connections between the steps

for each step x
  for each input item
    for each step y
      if step x is not step y
        for each output item
          if input item equals output item
            step x depends on step y
```

Fig. 7. (*Continued*)

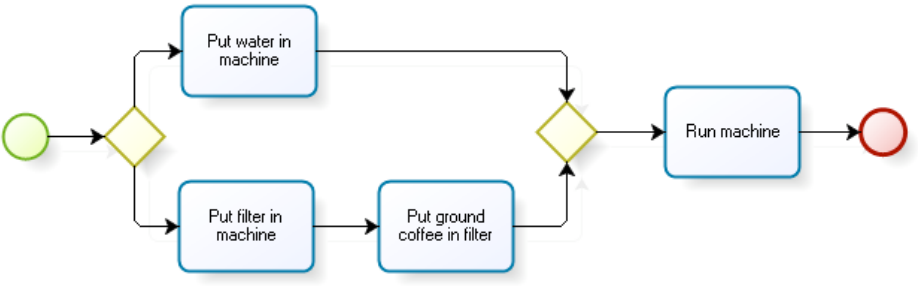

Fig. 8. Generated BPMN Diagram (implemented only as XPDL)

4 Development and Evaluation

The development process was performed within the design science paradigm [15], in a standard yet admittedly somewhat ad-hoc development cycle involving design, implementation, testing and improving the design. Testing was not always done extensively in the initial development stages, but as the game evolved from little more than a list of rules to a digital, more graphical game, more systematic evaluation took place.

We initially tried to stay away from automating the game, focusing instead on a pen and paper approach ("board game"). This gave us the chance to think through the basic setup and rules, without getting lost in details of implementation. However, the down sides of board gaming also became apparent soon. Playing and therefore testing the game proved to be a tedious experience for both the facilitator and the player, mostly because of active relations and dependencies between items, which had to be updated manually.

The first digital version of the game soon appeared, based on a standard spreadsheet implementation, which at least took the task of calculating the score out of the

hands of the facilitator, and also made the entering of information by the player much easier. This allowed us to perform the first (successful) tests with “outsiders” (i.e. players other than ourselves), proving that the game system was viable in principle, and that players could at least get through the game.

It became increasingly clear at this point that the expectations for and experience of playing an actual computer game rely not just on its rules, but also on flowing interaction, animation and sound. The choice was made to rebuild the game from scratch in Actionscript 3.0 [16], which, after a few iterations, led to the current prototype.

The game was tested on five players with little to no prior process modelling experience, as well as on five players with significant such experience. Each test player played the game three times. The first two times, simple, standard tasks were described: first, "making coffee" and second, "repairing a flat bicycle tyre". These are tasks everyone in the test population was familiar with, the second generally being a little more complicated than the first. The third game was played with a task of the player's own choice.

The players were given as little introduction as possible, forcing them to rely on the explanation provided by the game. Admittedly, some subtle hints were sometimes given during the game if a player was really stuck. The players were asked to ‘think aloud’ as much as possible and to voice their possible frustrations or confusion. The attending game developer wrote down observations and interesting comments.

Afterwards the players were given a questionnaire, consisting of 25 statements (with five possible responses, ranging from “strongly disagree” to “strongly agree”) and three open questions. The purpose of the questionnaire was to get an idea of the players' general experience. The statements hardly go into aspects unique to this particular game. Examples are: "I was satisfied with the result when I was done" and "I felt the score was a fair representation of how well I was doing". The statements are based on known properties of successful games, described in game design literature [11,17,18] and on general usability heuristics [19].

We aimed for our observations and player comments, as well as the results of the questionnaire, to inform us about the differences between players with and without process modelling experience, with respect to success in and perception of the game. We also looked for differences in player perception when playing the game for the second or third time. In general, we were obviously also interested in whether or not the game was properly designed, and in particular in needs or wishes for further improvement. Note that our game was a very first attempt of creating a game-for-modelling, so finding out what does *not* work was expected to be a prominent part of our effort.

5 Lessons Learned

Our observations suggest that there is indeed a noticeable difference between players with and without modelling experience. Despite our intentions to make the game playable for players with little or no expertise in modelling, most of those players found it hard to get started. It took them a while to understand what was meant by “task”, “step”, “ingredient” and “product”. It was somewhat of a surprise to us that people by nature do not seem to make a sharp distinction even between actions and

objects: sometimes they confuse the name of a step with its products, or describe sub-steps instead of ingredients.

Generally the game has a hard time forcing people towards the 'correct' way of thinking, if they are not inclined that way already. For example, while the game is supposed to be played by listing steps, some people naturally start describing a task by listing ingredients. They even search for workarounds to do this, like making one step a super-step encompassing the whole task, or describing a first step 'fetch all ingredients'.

Most experienced players fare better, quickly grasping the concept of the game, although sometimes after some initial confusion. In contrast with the less experienced players, they list ingredients only in their head, and then quickly switch to describing steps. Not surprisingly, experienced players are also more conscious of issues pertaining to abstraction.

When it comes to learning to play the game, the one thing we can clearly identify is the moment that players really 'get the hang of it'. Experienced players generally start working quickly and efficiently near the end of the first game or at the beginning of the second, while less experienced players are still struggling during the second game, and only pick up pace during the third. On the up side, given that playing the game does not take all that much time and effort, this could still be considered reasonably rapid learning.

A further interesting observation is that advanced functions of the game are generally not used. Players simply look for the easiest way to succeed in the game. Only one player so far has used preconditions in his description.

The results of the questionnaire suggest that experienced players found it more important to play well, while the inexperienced players were more satisfied with their results. All players felt there was a trick to easily getting a high score, but still mostly thought that the score was a fair representation of how well they did. This implies that the "trick" (whatever it was) was not actually used.

There was very little variation in how easy players thought it was to get started (most answers were neutral), but surprisingly the experienced players were on average less satisfied with the amount of context-sensitive help.

The first open question of the questionnaire ("What was, according to you, the most important goal in the game?") turned out particularly interesting because it really touches on the conflict between game-internal and game-external goals that we have been dealing with. The question is vague on purpose and results in a variety of answers, such as: "scoring points" (pure game-internal), "finding out what the links between steps are" (basic game-external), and "judging how well people can model" (extreme external: concerning reflection on the modelling process).

5.1 What Went Well

It is encouraging that players generally do not take very long to learn how to play the game and how to produce a reasonable task description. It seems that the main strength of the game is that it provides players with tangible feedback based on what they are doing, giving them at least a general sense of direction in the modeling process. This is an improvement over having to explain goals of modelling in abstract terms to people with little process modelling experience.

Aside from a few specific problems, the players seem to be satisfied with the current interface. By keeping things simple it manages to provide a good overview of the whole task description within one screen, and the feedback it provides by using graphics and sound seems to work well.

5.2 What Went Not So Well

We cannot honestly say that the current score system properly represents the various quality aspects of modelling. Players recognized that they did not really have to provide sensible input to get a good score. There is certainly much room for improvement here, although quantifying quality will never be easy.

It is somewhat disappointing that even though the scope of the game was already limited, most players did not use the advanced functionality such as preconditions. Players may be better encouraged to do this if provided with examples of more complicated descriptions, but we avoided the use of examples so far. Instead we used a wizard-like system to get players started, which is easy to understand, but is limited when things get more complicated.

A more specific problem we encountered concerns the use of links between ingredients and products within a step. If preconditions are not used, the purpose of these links is unclear to players and they consistently tend to forget about them. In fact, there are no clear rules concerning when links are required, so the confusion is entirely understandable. This is something that still requires attention.

6 Conclusions and Further Research

We set out to demonstrate and discuss a very first prototype of a "game for modelling", which is in fact a first implementation of the idea of a "modelling wizard". We explained the rationales behind the idea of creating modelling games/wizards, and proceeded to describe the basic utilitarian purposes (goals) behind our prototype game: deriving some basic information from a game with the purpose of then generating a basic BPMN diagram, focusing only on the derivation of AND-splits and joins.

We then described the game as such, and the process of developing and evaluating it. We finished with the outcome of our evaluation, observations, and lessons learned. In particular we concluded that though the game worked reasonably well, it is certainly not "intuitively playable" at this point. Arguably, this means we failed to achieve our main goal. However, this being a first attempt, we are by no means discouraged, and believe we have demonstrated what a game-for-modelling might look like and –in some respects– what it should perhaps not look like. In the mean time, we consider our first proof of concept a modest success: the game exists, is playable, renders sufficiently usable results, and is considered moderately satisfying by the players. It now serves as a platform from which further explorations can depart.

Whether modelling can ever be remotely as much fun as a dedicated entertainment game remains to be seen, but we can certainly learn a lot in this respect from how games are designed. A good game is more than just a goal to work towards. It lets you know what interactions are at your disposal, it lets you explore what effect your actions have on the game world, and it gives you a sense of how well you are doing.

Similarly, a good modelling process requires more than just a language and a model to work towards.

Future research will, of course, focus on further development of the game. We consider various directions for doing so. Improvement of this actual prototype is a possibility, but we are also interested in developing a more strongly text-oriented version. We also consider expansion of the concepts to be elicited by the game, possibly by combining a number of sub-games (covering different modeling rationales). A PhD project has recently started at our department which aims at the creation of a larger, much more developed game, better rooted in theory and practice involving game design theory and game psychology, cognition (in particular pertaining to abstraction and conceptualization), but also AI (reasoning about information gathered, after the game but also during the game).

Evaluation of games played is a key aspect of the development cycle, and requires extra attention in any effort that claims to engage in "design science". Another ongoing PhD project [20] focuses on advanced evaluation of interactive modelling sessions (including both collaborative and solo modelling), and methodological results will be used to evaluate games played.

References

1. Ralyté, J., Brinkkemper, S., Henderson-Sellers, B.: Situational Method Engineering. In: Fundamentals and Experiences. IFIP, vol. 244, pp. 313–327. Springer, Boston (2007)
2. Hoppenbrouwers, S., van Bommel, P., Järvinen, A.: Method Engineering as Game Design: an Emerging HCI Perspective on Methods and CASE Tools. In: Exploring Modelling Methods for Systems Analysis and Design (EMMSAD 2008), held in conjunction with CAiSE (2008)
3. Krogstie, J., Sindre, G., Jorgensen, H.: Process models representing knowledge for action: a revised quality framework. European Journal of Information Systems 15, 91–102 (2006)
4. Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H.A., van der Aalst, W.: Quality metrics for business process models. In: Fischer, L. (ed.) BPM and Workflow Handbook. Future Strategies and Workflow Management Coalition (2007)
5. Lagerstrom, R., Saat, J., Franke, U., Aier, S., Eckstedt, M.: Enterprise Meta Modelling Methods: Combining a Stakeholder-Oriented and a Causality-Based Approach. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modelling. LNBIP, vol. 29, Springer, Heidelberg (2009)
6. Proper, H.A., Verrijn-Stuart, A.A., Hoppenbrouwers, S.J.B.A.: Towards Utility-based Selection of Architecture-Modelling Concepts. In: Hartmann, S., Stumptner, M. (eds.) Proceedings of the Second Asia-Pacific Conference on Conceptual Modelling (APCCM 2005), Newcastle, New South Wales, Australia, January 2005. Conferences in Research and Practice in Information Technology Series, vol. 42, pp. 25–36. Australian Computer Society, Sydney (2005)
7. Faget, J., Marin, M., Mégard, P., Owens, V., Tarin, L.: Business Processes and Business Rules: Business Agility Becomes Real. In: Workflow Handbook 2003, pp. 77–92. Future Strategies Inc. (2003)
8. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modelling Language User Guide. Addison Wesley, Boston (1998)
9. Object Modelling Group: Business Process Modelling Notation (BPMN) version 1.0, OMG Final Adopted Specification (2006)

10. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems 30(4), 245–275 (2005)
11. Järvinen, A.: Games without Frontiers, Theories and Methods for Game Studies and Design. PhD Thesis, University of Tampere, Finland (2008)
12. Hoppenbrouwers, S.J.B.A.: Community-based ICT Development as a Multi-Player Game. In: What is an Organization? Materiality, Agency and Discourse. University of Montreal (2008)
13. van Bommel, P., Hoppenbrouwers, S.J.B.A., Proper, H.A., Roelofs, J.: Concepts and Strategies for Quality of Modelling. In: Innovations in Information Systems Modelling: Methods and Best Practices. IGI, New York (2008)
14. Workflow Management Coalition: XPDL 2.1 Complete Specification (Updated October 10, 2008)
15. Hevner, A.R., Ram, S., March, S.T., Park, J.: Design Science in Information Systems Research. MIS Quarterly 28(1), 75–105 (2004)
16. Adobe Labs, ActionScript 3.0 specification, http://labs.adobe.com/technologies/actionscript3/
17. Salen, K., Zimmerman, E.: Rules of Play, Game Design Fundamentals. MIT Press, Cambridge (2004)
18. Desurvire, H., Caplan, M., Toth, J.A.: Using heuristics to evaluate the playability of games. In: CHI 2004, Vienna, Austria, extended abstracts on Human factors in computing systems, pp. 1509–1512. ACM, New York (2004)
19. Nielsen, J.: How to Conduct a heuristic Evaluation. Online paper, http://staff.unak.is/not/nicolaw/courses/hci/HCILab7papers.pdf
20. Ssebuggwawo, D., Hoppenbrouwers, S.J.B.A., Proper, H.A.: Analyzing a Collaborative Modelling Game. In: Yu, E., Eder, J., Rolland, C. (eds.) Proceedings of the CAiSE 2009 Forum at the 21st International Conference on Advanced Information Systems Engineering, Amsterdam, The Netherlands, June 8-12 (2009) Published online (May 28, 2009), http://CEUR-WS.org/Vol-453/

View publication stats